

**Getting Started with Microsoft Visual C++**  
A Tutorial in 2 or 3 Acts  
By M. Morrison

## 1.0 Scope

This tutorial demonstrates the basic steps required to create and execute C++ programs using Microsoft's Visual C++<sup>®</sup> software package. The target audience is students enrolled in Salisbury State University's COSC120 computer science course. It is assumed that students are familiar with the Windows 95<sup>®</sup> operating environment and know how to logon to the university's computer system.

## 2.0 Introduction

Visual C++ is a very powerful package but, as is usually the case with powerful software, it is also very complex. This tutorial illustrates one of the simplest ways to get started creating programs using Visual C++; it is by no means the only way. This tutorial is broken down into 3 parts. Section 3.1 describes how to work with existing C++ source code files as is done in the early course labs. Section 3.2 describes how to create new programs from scratch. Section 3.3 (once written) illustrates use of the Visual C++ debugger. General operations such as saving and printing files are covered in Section 3.1.

This tutorial assumes that the reader will be using a machine in one of the SSU computer labs. The figures in this tutorial are screen shots from an actual programming session. Since there is usually some variation among lab computers, it is unlikely that the screen shots in this tutorial will correspond in every detail with what you see on your own screen. However, the essential characteristics are the same. The procedures outlined here should also be applicable to readers using their own computers. The location of files will be different, but the tasks are the same.

### 3.0 Programming with Visual C++

The following subsections take you step-by-step through actual programming sessions. Section 3.1 shows how to open, compile, execute, save, and print an existing file. Section 3.2 describes how to create and debug new programs and assumes that the reader is familiar with the material in Section 3.1. Section 3.3 (once written) illustrates simple debugger operations and assumes that the reader is familiar with the material in Sections 3.1 and 3.2.

#### 3.1 Working With Existing Files

The following paragraphs describe the steps necessary to open, compile, execute, save, and print an existing C++ source code file. A source code file is a file that contains C++ programming statements and usually has a file type or extension of ".cpp." For example, `hello.cpp`.

1. Open Visual C++ from the Windows Start menu by selecting **Programs | Microsoft Visual Studio 6.0 | Microsoft Visual C++ 6.0**. See Figure 1 below.

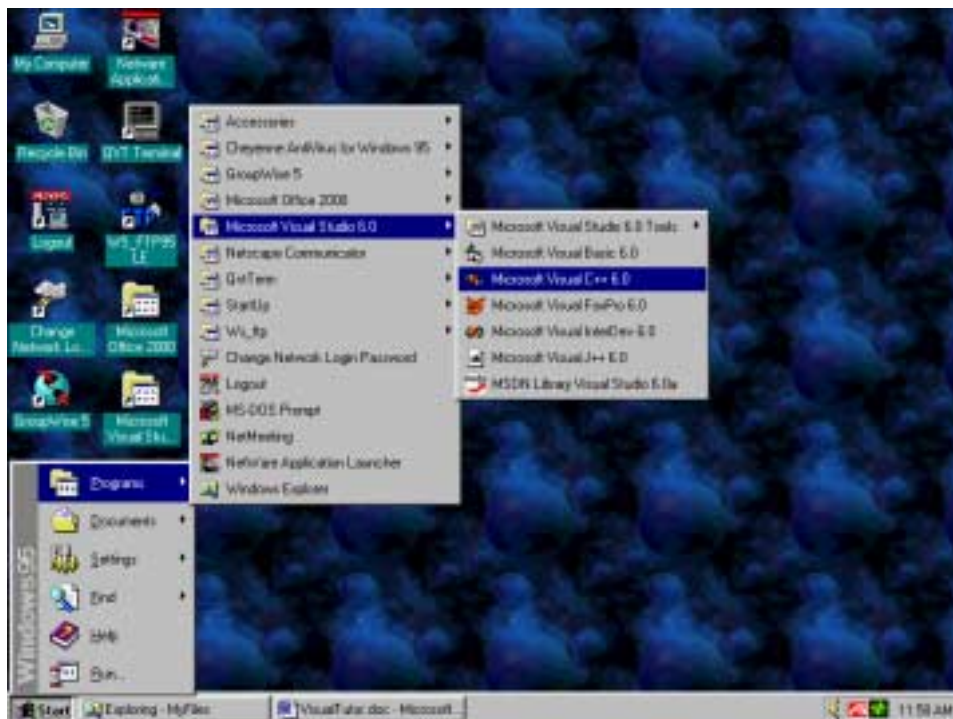


Figure 1. Opening Visual C++ from the Windows Start menu.

2. From the Visual C++ **File** menu, select **Open**.
3. Navigate to the proper directory (or folder). Files for labs will normally be located on the `K:` drive, also named `School` on `'Fs_apollo\User'`. In this tutorial, we will open the file `Firstone.cpp`. This file is associated with the first COSC120 lab and is located in the `Henson\COSC\COSC 120 labs\Chap1` directory of the `K:` drive. Select

Firstone.cpp and then click on **Open**. Alternatively, just double click Firstone.cpp. See Figure 2.

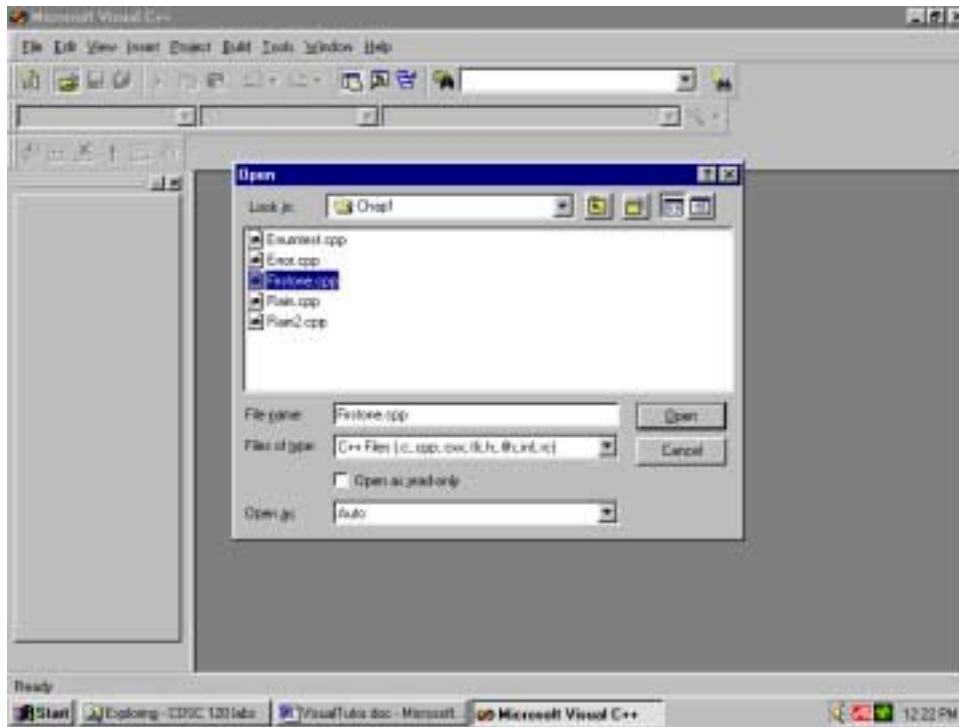


Figure 2. Opening the file Firstone.cpp.

4. The file will be opened in the Edit window inside the main Visual C++ window. At this point in our programming session, the main Visual C++ window is divided into two smaller windows (or panes). Source code files are opened in the Edit window. To the left of the Edit window is an empty gray pane. This pane is for the Class Browser window but we will ignore it for the remainder of this tutorial. The Firstone.cpp file can be resized to fill as much of the Edit window as you like.
5. The next task is to compile Firstone.cpp, but Visual C++ attempts to save a file before compiling it. Because we have opened one of the lab files, we can not save it to the same directory that we opened it from. This prevents us from accidentally overwriting the original version of the file with a changed version that we have worked on. You are advised to save files to your local C: drive. **NOTE:** You probably do not have adequate space on your network P: drive to build programs in Visual C++. For the time being, you will be much safer saving files to the local machine's C: drive. If you do save files to your local C: drive, you may wish to copy the files from the local machine to your P: network drive or to a floppy before logging off the lab computer. Once you log off, all the files you created during your programming session will be deleted from the local machine.
6. Do not save your source code file to the floppy (A:) drive from Visual C++. Visual C++ creates many new files of its own when it compiles and executes your source code file. If you save your source code file to a floppy, Visual C++ will try to save all its own files to the

floppy as well. A floppy disk is too small to hold all the files created and the result will be an error – your program will not execute.

7. Save `Firstone.cpp` to the `P:` drive by selecting **Save As...** from the Visual C++ File menu. Note that there is both a **Save** and a **Save As...** option on the File menu. You must choose **Save As...** See Figure 3.



Figure 3. Saving `Firstone.cpp` to a user directory.

8. Compile `Firstone.cpp` by selecting **Compile** from the **Build** menu. There may also be a compile icon on one of the toolbars. The icon contains a single arrow pointing down onto a stack of papers. If visible, it will usually be located 3 icons to the left of the red (or gray) exclamation mark icon. Clicking on the compile icon is equivalent to selecting **Compile** from the menu. See Figure 4.

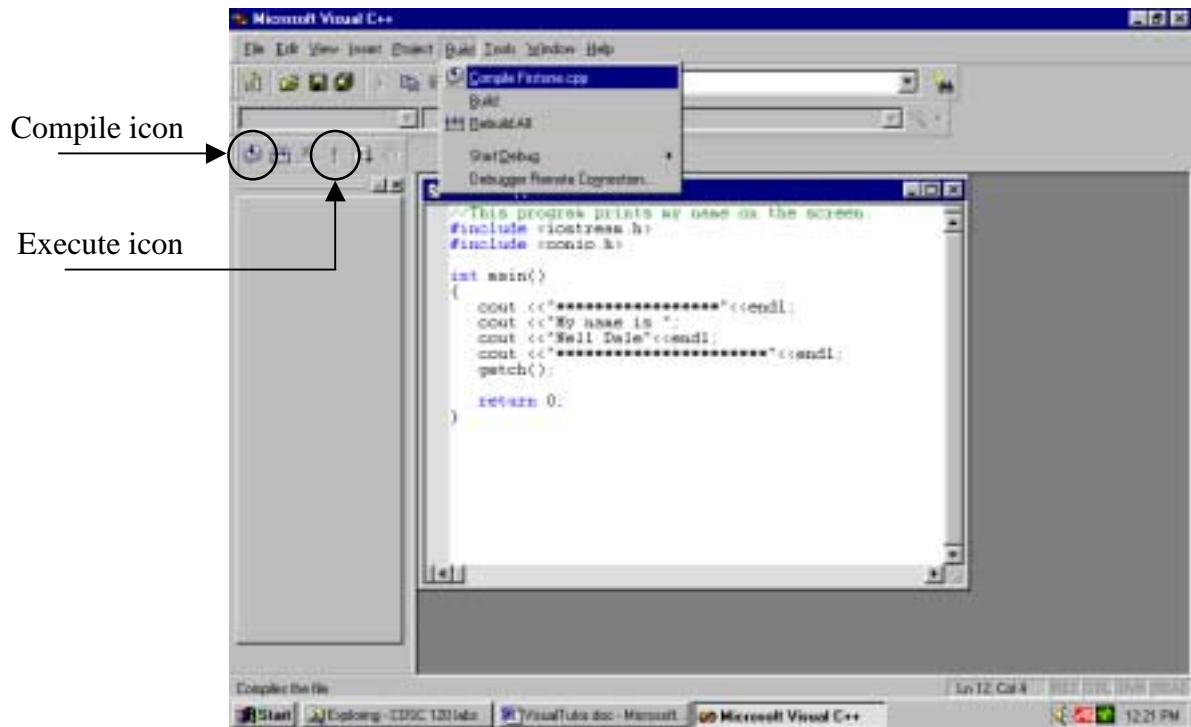


Figure 4. Compiling a source code file

8. The system will display a dialog window asking you to create a default project workspace. A project workspace contains all the Visual C++-created files discussed in paragraph 6. Select the **Yes** button in this window (Figure 5).



Figure 5. Create project workspace dialog window.

9. The program should compile without errors. If not, refer to the debugging paragraphs in Section 3.2 of this tutorial. At this point a third pane will open at the bottom of the main Visual C++ window. We will refer to this third pane as the Status window. It should reflect the fact that our program compiled without errors. If there were errors, the Status window will tell us where in the source code file those errors occurred. Figure 6 reflects the outcome of a successful compile.

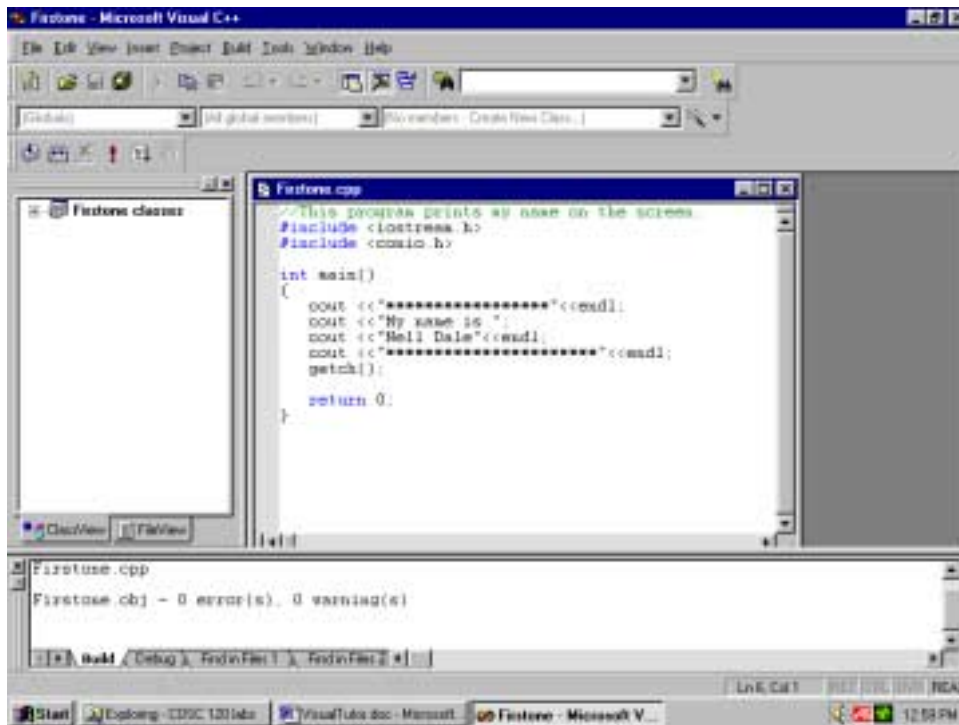


Figure 6. Status window reflects successful compile.

10. To execute the program, select **Execute** from the **Build** menu or click on the red exclamation mark icon shown in Figure 4. The system should display a dialog window asking you if you want to build `Firstone.exe`. Select the **Yes** button in this window (Figure 7).

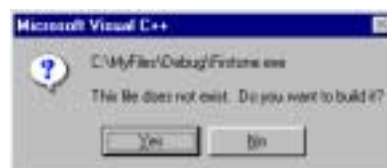


Figure 7. Build executable file dialog window.

11. The program will execute in a separate window as shown in Figure 9 below. This window is a MS-DOS console window. This window could appear almost anywhere on your screen. It is possible for this window to become obscured by other windows on your screen, including the Visual C++ window itself. If you suspect this may have happened, check the Windows taskbar at the bottom of the screen. If your program is running in a console window, there will be an icon for it in the taskbar. Look for the stylized MS-DOS logo as shown in Figure 8.



Figure 8. Console window taskbar icon.

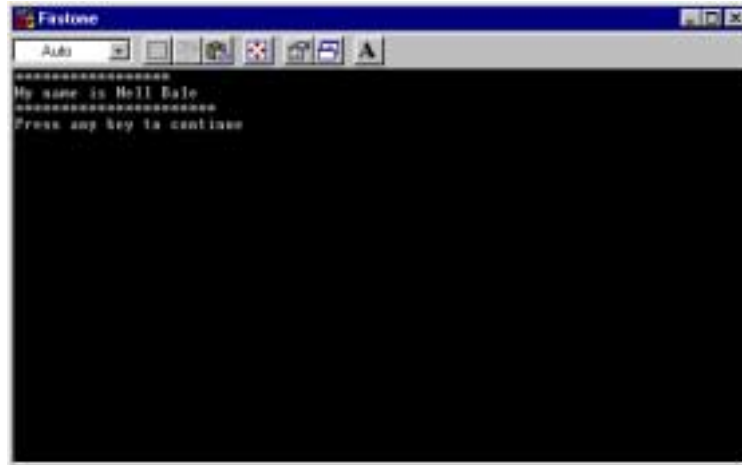


Figure 9. Program output in MS-DOS console window.

12. It is also possible for several of these windows to be open simultaneously; this means there are multiple copies of your program running at the same time. It is easy to get confused, especially when debugging a program. Be sure to close each console window after running your program. If your program freezes and won't do anything no matter what key you press, close the console window by clicking on the 'X' icon in the upper right hand corner of the window.

You may occasionally get the following error in the Status window when trying to compile and execute your program:

```
LINK : fatal error LNK1168: cannot open
Debug/Firstone.exe for writing
Error executing link.exe.
```

This means that there is already a copy of your program running. Visual C++ can not create the new version of your program while the previous version is still running. Close the currently running version of your program and then rebuild.

13. Print the source code file by selecting **Print** from the Visual C++ **File** menu. When the **Print** dialog window opens, click the **OK** button. The exact details of printing files in the labs will vary depending on which lab you are in. Your best bet is to accept the default printer settings wherever you are. It may be possible to select specific printers as shown in Figure 10. Odds for success are good if the room number of the lab you're in appears in the printer name. Figure 10 shows that the selected printer is in PP108; room 108 in the Power Professional building.
14. To quit Visual C++, select **Exit** from the **File** menu. If you have made any changes to your source code file since it was last saved, Visual C++ will prompt you to save it before exiting. Make sure you have saved your files in the appropriate locations. Anything you save to the lab machine's local C: or D: drives will be wiped out once you logout.

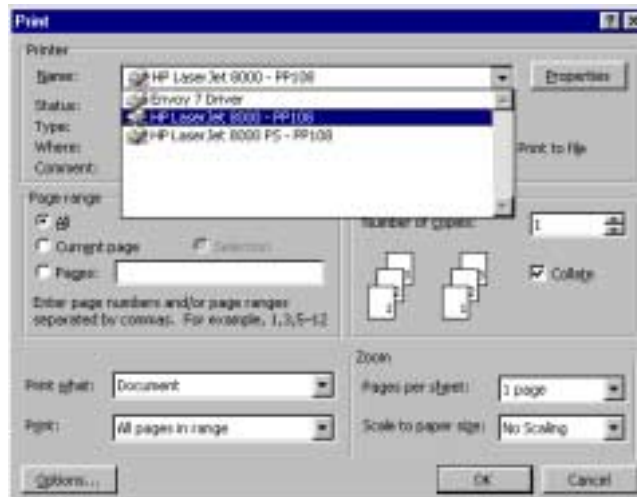


Figure 10. The Print dialog window.

15. You must log off the computer when done working. Failure to do so allows someone to come behind you and use the computer under your user name. You are responsible for anything that happens to the system that can be traced to your user name.

### 3.2 Creating a New Program

The following paragraphs describe the steps necessary to create, compile, debug, and execute a new C++ source code file. This section of the tutorial assumes that you are familiar with the operations covered in Section 3.1.

1. Start Visual C++.
2. From the Visual C++ **File** menu select **New...**  
Note: Use the **File** menu and do not click the New File icon on the toolbar.
3. Refer to Figure 11 during the following steps.
4. When the **New** dialog window appears, select the **Files** tab.
5. Click (do not double click) on the **C++ Source File** option.
6. In the **File** name box, enter the desired name of the source code file. In this tutorial we will name the file `NextOne`. Note that it is not necessary to give the file extension when entering the file name in this dialog window. Visual C++ will automatically assign the `.cpp` extension.
7. In the **Location** box, enter the name of the directory or folder where the source code file should be saved. In this tutorial we will save our source code file to the local machine's

C: drive. Refer to paragraph 5 of Section 3.1 for a discussion of preferred file locations. Make sure there is plenty of free space on the drive where you save the source code file. Visual C++ will create many support files when it compiles your program and it will place all these extra files in the same directory that it places your source code file.

Note: Since we have saved the source code file to a local drive rather than to our network drive, we absolutely must save the source code file to a floppy or our network drive before logging off the computer. It is not necessary to save all the extra files that Visual C++ creates. Visual C++ will recreate these files the next time the source code file is compiled and executed.

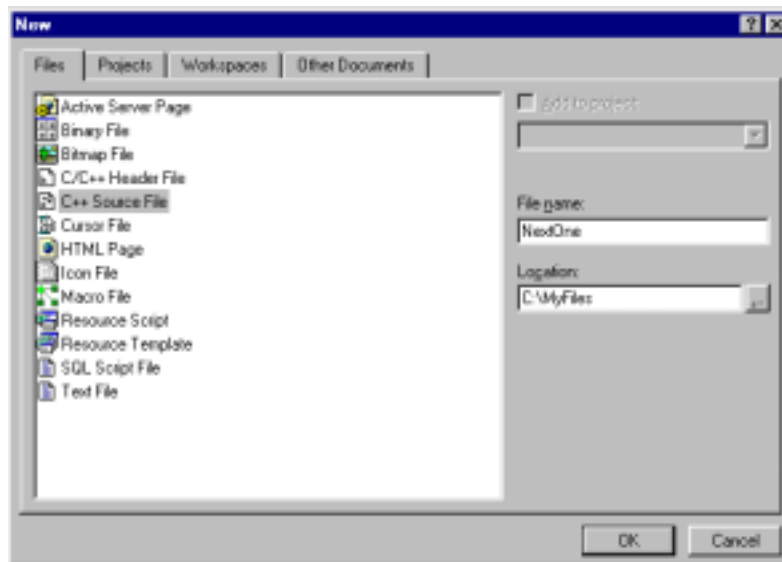


Figure 11. The New File dialog window.

8. A new, empty file is opened in the Visual C++ Edit window ready for us to add our code. Enter the following program as written. Note that the program contains some intentional errors.

```
//.. This program is the classic Hello, world
//.. modified for use in a simple tutorial.

include <iostream>
using namespace std;

int main()
{
    cout << "Hello, world!"

    return 0;
}
```

9. Compile the program.

10. Visual C++ will display a dialog asking if you want to create a default project workspace. Say yes.
11. Visual C++ will display a dialog asking you whether to save the changes made to `NextOne.cpp`. Say yes.
12. The Status window at the bottom of the main Visual C++ window should report 6 errors. Don't panic; many of them are related. It is often the case that a single syntax error will result in multiple error messages from the compiler.
13. It is possible to resize the Status window by locating the cursor on the top border of the Status window. When the cursor changes to a pair of parallel lines with arrows attached, you can click and drag the window border to make the window larger or smaller.
14. Scroll to the top of the Status window so that the first error message is visible. By double-clicking on the first error message, a small blue arrow in the left margin of the Edit window will point to the line in the source code where the compiler detected the error. See Figure 12 below.

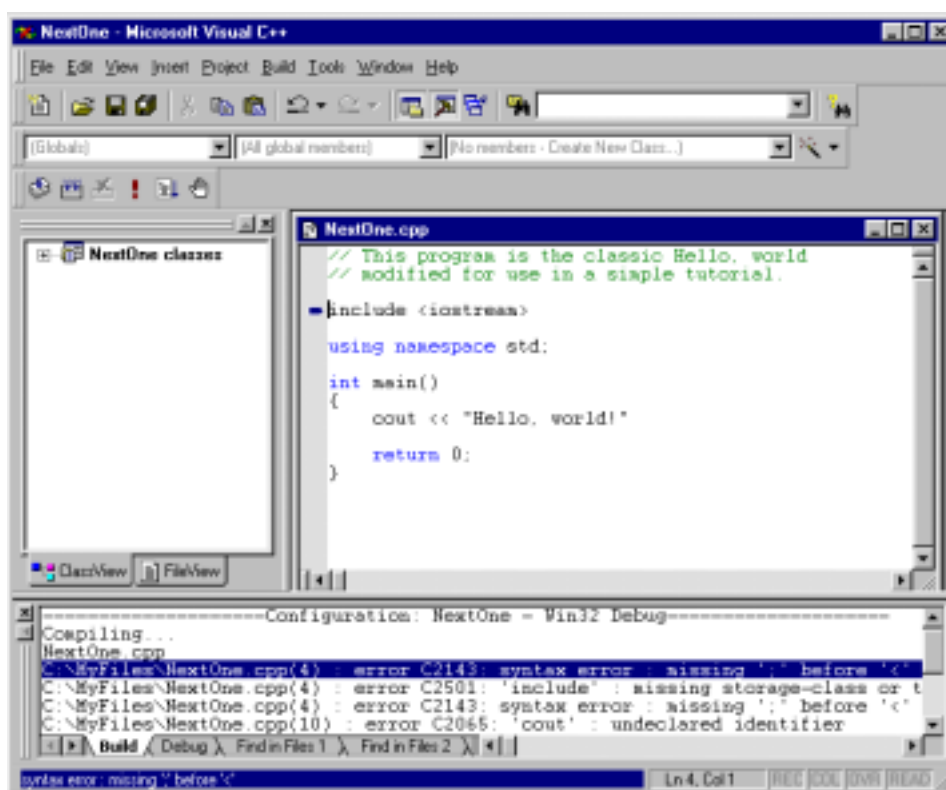


Figure 12. First error location.

15. In this example, the error is the missing '#' character before `include`. Note that the next two errors are on the same line and are related to the first error. The "(4)" in the first three error messages indicates that the errors all occurred on (or near) the fourth line of the program. In this case, the compiler generated at least three error messages for one syntax error.

16. Correct the indicated error in the source code. Correcting the error does not immediately make the error messages in the Status window disappear. They will not change until the file is recompiled.
17. Scroll down to the fourth error in the Status window. The compiler is telling us that the word `cout` is an undeclared identifier in line 10 of our program. As far as we know, `cout` is a perfectly legal word to use in this context – we use it all the time in class just like this. The word `cout` tells the compiler we’re trying to display something on the screen; we’re trying to do some kind of output. But we can only do output if we first `#include` the `iostream` library and the line of code that tried to do the `#include` contained an error. Because of that error, the compiler never saw the `iostream` library so it never knew about `cout`. Since this is yet another manifestation of the first error, we can ignore it. What you shouldn’t ignore is the ripple effect of syntax errors. If you get many error messages when compiling, you might find it easiest to just fix the most obvious errors, recompile, and see what’s left. Many errors may disappear on their own.
18. Double click on the fifth error that says “<<” is illegal. See Figure 13. The error pointer is indicating that the problem is in the `return` statement of our source code file, but there is no “<<” in the `return` statement. This makes no sense to us so we check the next error in the list. We see that the next error is in the same line of the file (line 12) and mentions a missing semicolon. Scroll the Status window to the right so that the rest of the error message can be read. The complete error message tells us that the missing semicolon should come before the `return` statement.

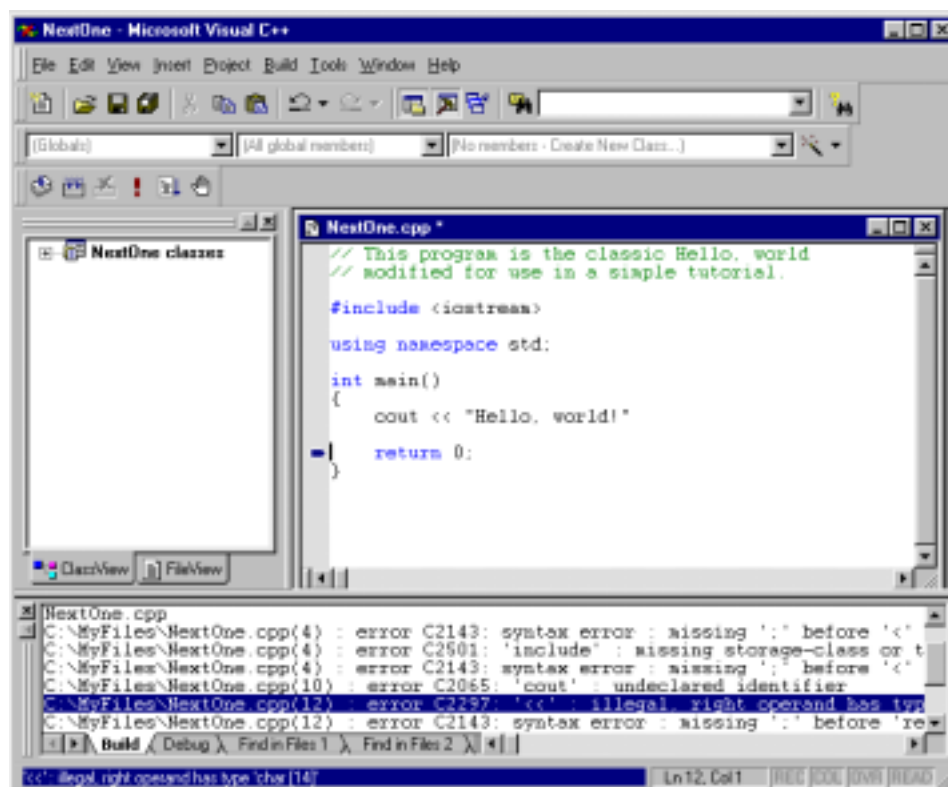


Figure 13. Second error location.

This tells us that the real error is not in line 12 but in the line before: line 11. Line 11 is missing its final semicolon.

Important Tip: If you can not find an error in the line indicated by the error message, check the line immediately before it.

19. Add the missing semicolon to line 11.
20. Recompile the program. It should compile error-free. If not, you know what to do...
21. Execute the program. You will see the dialog asking about building `NextOne.exe` again. Say yes.
22. The program should appear running in a console window.
23. From this point you can edit, save, or print the source code file. If copying your program to another drive or a floppy, you do not need to copy any file except the `.cpp` source code file. The other files are created by Visual C++ and Visual C++ will recreate them the next time you build your program.